SOLAR: SPARSE ORTHOGONAL LEARNED AND RANDOM EMBEDDINGS

Tharun Medini, Rice University

Beidi Chen, Stanford University

Anshumali Shrivastava, Rice University

This work proposes SOLAR, a high-dimensional and ultra-sparse embedding learning method, which is a significantly superior alternative to dense low-dimensional embedding for both query latency and accuracy in Search Engines.

Unique Design Choices:

- To facilitate trivial distribution across GPUs, we design label embeddings to be **super-sparse** and **orthogonal**.
- We **spread out** the non-zeros of label vectors uniformly across the high dimensional space and **fix** the label vectors and **only learn** the query vectors.

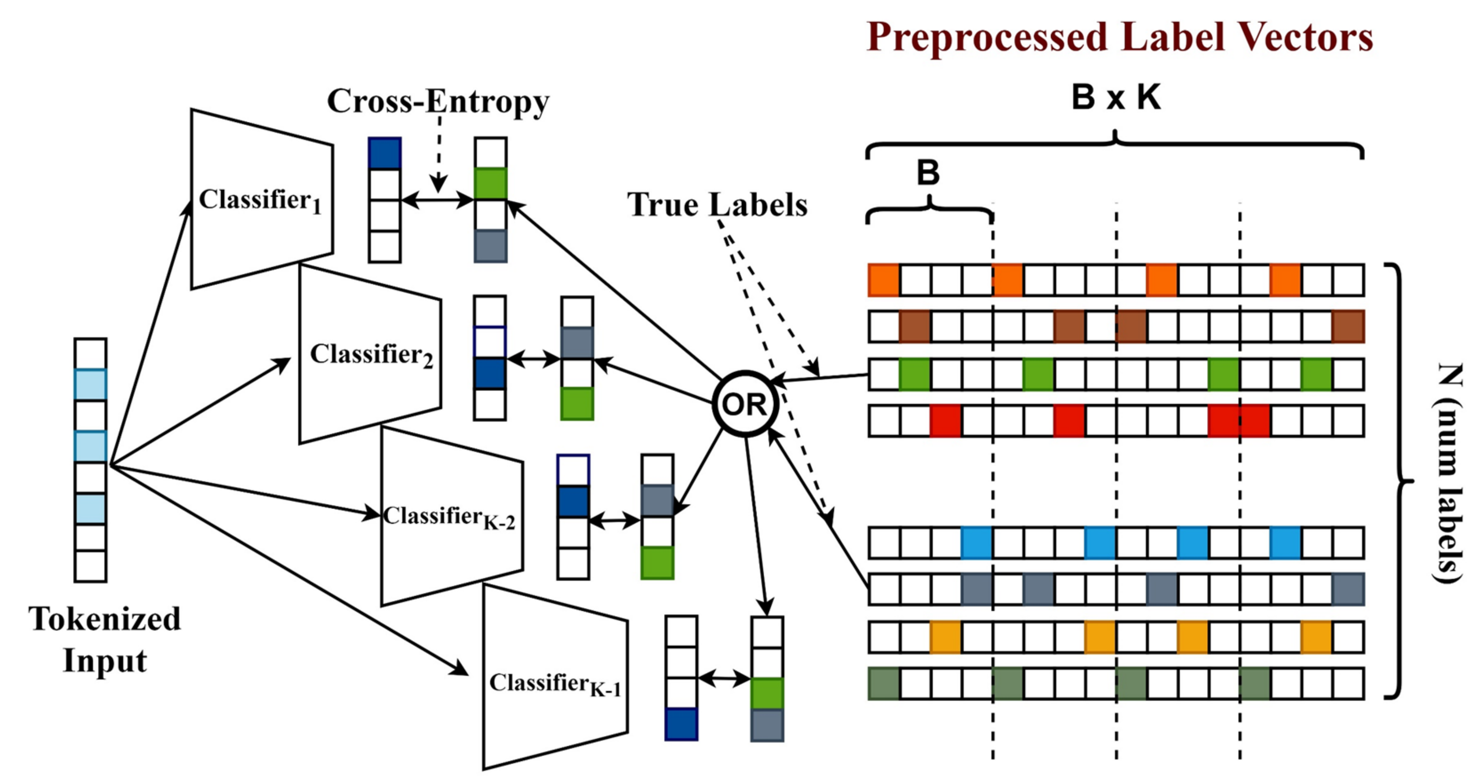
SOLAR has 4-fold advantage:

- Matrix Multiplication is replaced by cheap Inverted-Index Lookups
- Load-balanced Inverted Indexes
- Lower Embedding Memory
- Zero-communication distributed training of embeddings

Our Proposal: SOLAR

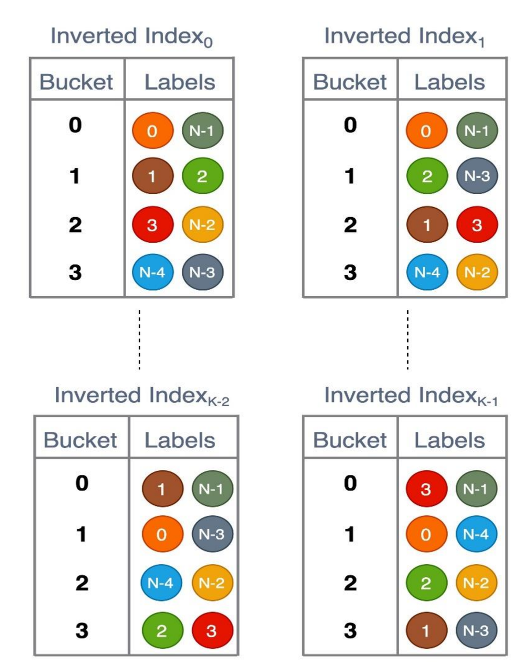
- **Notations:** **N** denotes the total number of labels. **D** is the sparse vector dimension. **K** is the number of non-zeros in label vectors. **B=D/K** is the number of buckets in each component of the vector.
- **Preprocessing:**
  - Partition **D** vector into **K** chunks
  - Each chunk has **B** buckets with exactly one non-zero index
  - The single non-zero index is **picked randomly** in the range of **B** for each of the **K** components
  - The dot-product between any two label vectors is **~0**
- **Training:**
  - Lookup all true label vectors for an input
  - Perform an **'OR'** operation over the respective sparse vectors
  - Partition the combined label vector into **K** chunks
  - Train **K** feed-forward networks to predict one each of the **K** chunks
- **Inference:**
  - Pass an input through all **K** models
  - Sort the **B** scores in each model to get **top-m** buckets
  - Query the **m*K** buckets in the inverted index and get the union of all candidate labels
  - Noisy candidates: Due to random initialization of label vectors, irrelevant labels are pooled together. We will omit all labels below a certain frequency threshold **t** across **K** models.
  - For each candidate, sum the predicted probability scores for the corresponding bucket and sort for the top results

Training



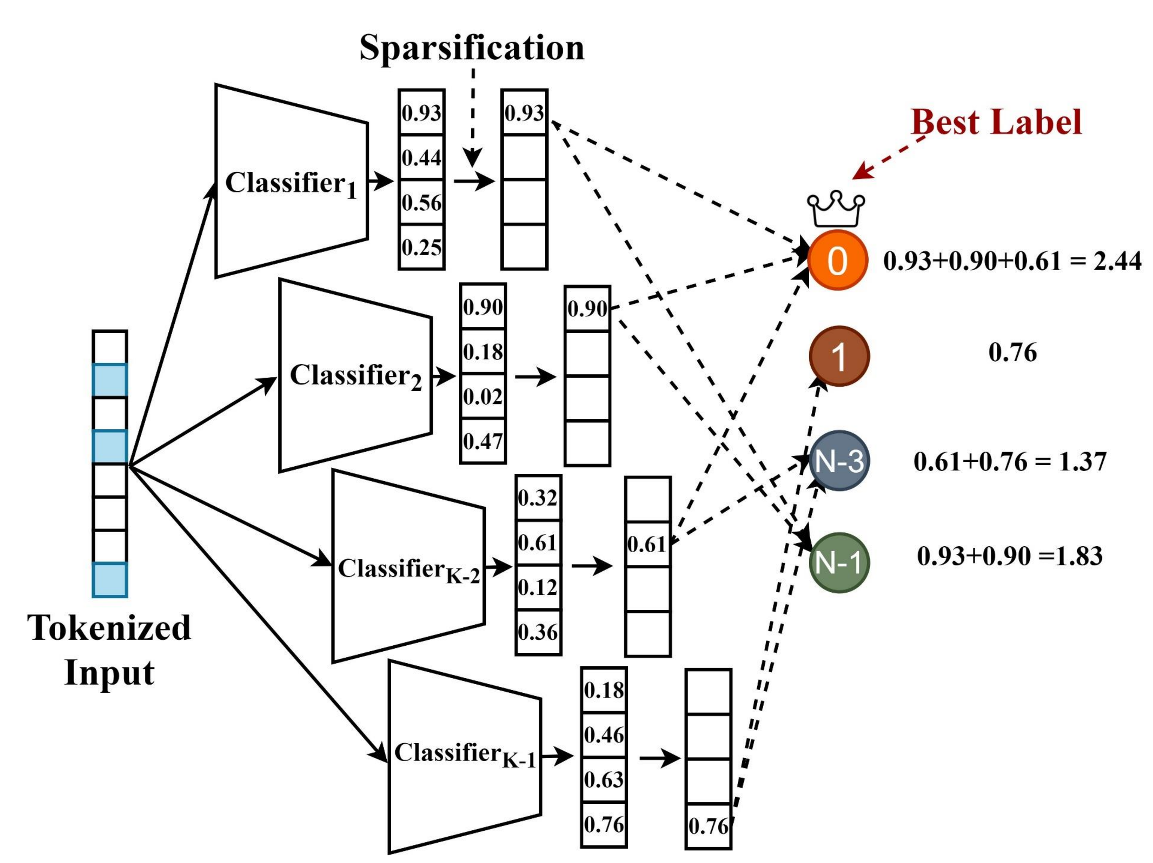
Schematic diagram for **label vector construction** (on the right) and the **training process** (on the left). Each label vector is **B*K** dimensional divided into **K** components of length **B**. Each vector is **K**-sparse with exactly one non-zero index in each component (colored on the right). The components are separated by dotted vertical lines. For a given input, we perform an **'OR'** operation over the true label vectors and feed the resultant pieces to independent small classifiers.

Inverted Index



Inverted-Index construction for the label vectors shown in the top figure. We construct one index for each of the **K** chunks. Each bucket will have the same number of labels by design (Load-Balanced)

Inference



Schematic diagram for Inference. We first get **K** probability vectors of **B** dimensions each. Then we only retain the **top-m** buckets after sparsification (**m**=1 in above figure. For our experiments, **m** varies among 50 and 100). We accumulate the candidate labels based on inverted-index for these top-buckets and aggregate their scores and identify the best labels

Product-to-Product Recommendation

Model	epochs	P@1	P@5	P@10	Rec@ 100	Train time (hrs)	Eval time (ms/point)
SOLAR (m=100)	10	35.24	29.71	26.98	34.19	2.65	0.96
DSSM (d=1600)	5	31.34	27.55	24.41	32.71	25.27	1.77
GLaS (d=1600)	5	32.51	28.31	25.41	33.17	37.14	1.77
SNRM (d=30K)	5	1.59	2.01	1.93	2.41	-	-
AnnexML (d=800)	10	26.31	22.22	19.37	26.13	16	3.06

Comparison of SOLAR against DSSM, DSSM+GLaS, and SNRM baselines. SOLAR's metrics are better than the industry-standard DSSM model while training **10x faster** and evaluating **2x faster** (SOLAR-CPU vs DSSM-GPU evaluation). GLaS regularizer improves the metrics but still lags behind SOLAR.

Extreme Classification Datasets

Dataset	Metric	SOLAR (m=100)	SOLAR (m=50)	Annex ML	SLEEC	FyC	Parabel	PfastreXML	SLICE
	P@1	60.92	60.52	56.81	30.86	46.86	59.34	55	59.89
Wiki-500K	P@3	46.94	45.56	36.78	20.77	31.29	39.05	36.14	39.89
	P@5	45.32	45.28	27.45	15.23	25.17	29.35	27.38	30.12
	P@1	34.37	34.19	26.36	18.77	24.47	33.93	28.51	37.77
Amz-670K	P@3	32.71	32.51	22.94	16.5	20.44	30.38	26.06	33.76
	P@5	32.55	32.46	20.59	14.97	17.13	27.49	24.17	30.7
	P@1	44.89	44.61	41.79	-	-	**47.51**	43.83	-
Amz-3M	P@3	42.36	42.08	38.24	-	-	**44.68**	41.81	-
	P@5	41.03	40.69	35.98	-	-	**42.58**	40.09	-

SOLAR vs popular Extreme Classification benchmarks. Embedding models AnnexML and SLEEC clearly underperform compared to SOLAR. SOLAR even outperforms the state-of-the-art non-embedding baselines like Parabel and Slice. **The gains in P@5 are particularly huge (45.32% vs 31.57%)**. SLEEC and SLICE do not scale up to 3M labels (corroborated on XML-Repo)

Dataset		SOLAR (m=100)	SOLAR (m=50)	SLICE	Parabel	PfastreXML
Wiki-500K	Training time (hrs)	2.52	2.52	2.34	6.29	11.14
	Eval (ms/point)	1.1	0.76	1.37	2.94	6.36
Amz-670K	Training time (hrs)	1.19	1.19	1.92	1.84	2.85
	Eval (ms/point)	2.56	1.58	3.49	2.85	19.35
Amz-3M	Training time (hrs)	5.73	5.73	-	5.39	15.74
	Eval (ms/point)	2.09	1.87	-	1.72	4.05

Training and Evaluation speeds against the fastest baselines

Contact

Tharun Medini: tharun.medini@rice.edu

Anshumali Shrivastava: anshumali@rice.edu

RUSH-LAB: rush.rice.edu